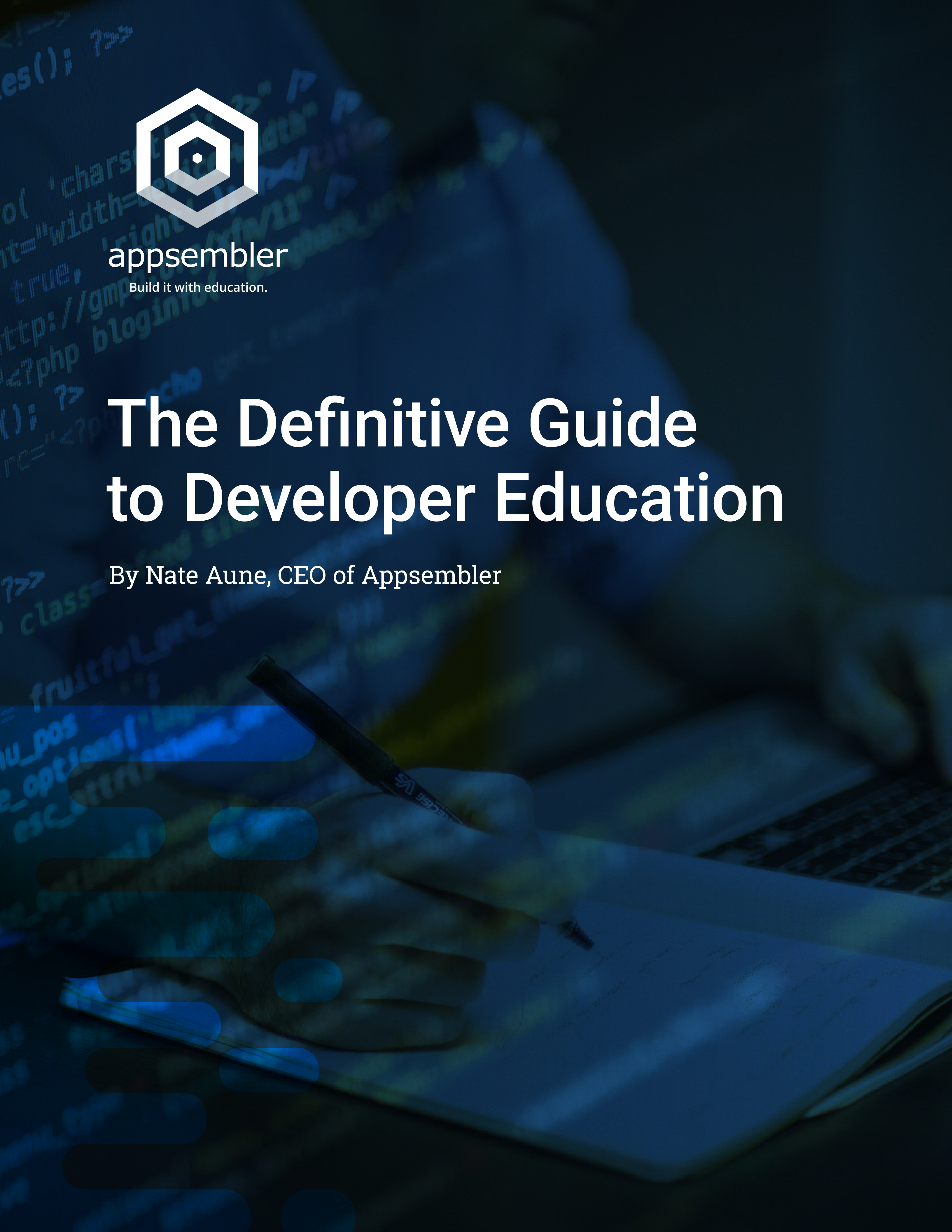


appsembler

Build it with education.

The Definitive Guide to Developer Education

By Nate Aune, CEO of Appsembler

There has been a 55% increase in the global population of software developers over the past four years. This growth is expected to continue with the global developer population expected to reach 28.7 million people by 2024, an increase of 3.2 million from 2020. These developers don't just write the code that creates products, they are also deciding which products they want their code to integrate and work with. There are now at least 1,000 companies whose products are primarily sold to developers and as of 2021, this developer-led economy has annual recurring revenues (ARR) of $49 Billion.

As a result, there are a lot more tools and products that are aimed at developers, and more companies who are selling their software directly to this market. Business-to-developer (B2D) companies are seeing massive growth, but they either survive or thrive based on their ability to get developers to use and adopt their products quickly. If they can't get developers interested and using their product, they will die.

These companies need to attract, nurture and convert developers to paying users, and then retain them as customers after the purchase. And once they are customers, these companies also want to expand their product's usage from a single developer to a team of developers, and perhaps eventually into the entire department. Developer education is a crucial tool in making this acquisition, monetization, and expansion process happen. It can help improve product awareness in various stages of the process and is integral to moving the developer through the purchase journey, from initial activation to long-term product engagement.

Why did we create this guide?

Developer education is an emerging space, and there aren't a lot of best practices about how to do it well, which is why we created this guide. It explains why developer education is so important, what you need to do to create a developer education program, and profiles the companies that have been successful at employing developer education.

Contents

Why is developer education so important?

Developers enjoy learning about technology, and they want to know about a product's features, rather than just read about its list of benefits. An effective program will educate developers on the product's features, help them understand how the product can solve their problems, nudge them towards product adoption, and increase their product usage.

But developers don't respond to traditional marketing techniques, such as cold calls, static sales demos, and marketing brochures. Because of this, the path to developer purchase — and all the steps along the way — looks different than the traditional purchasing journey.

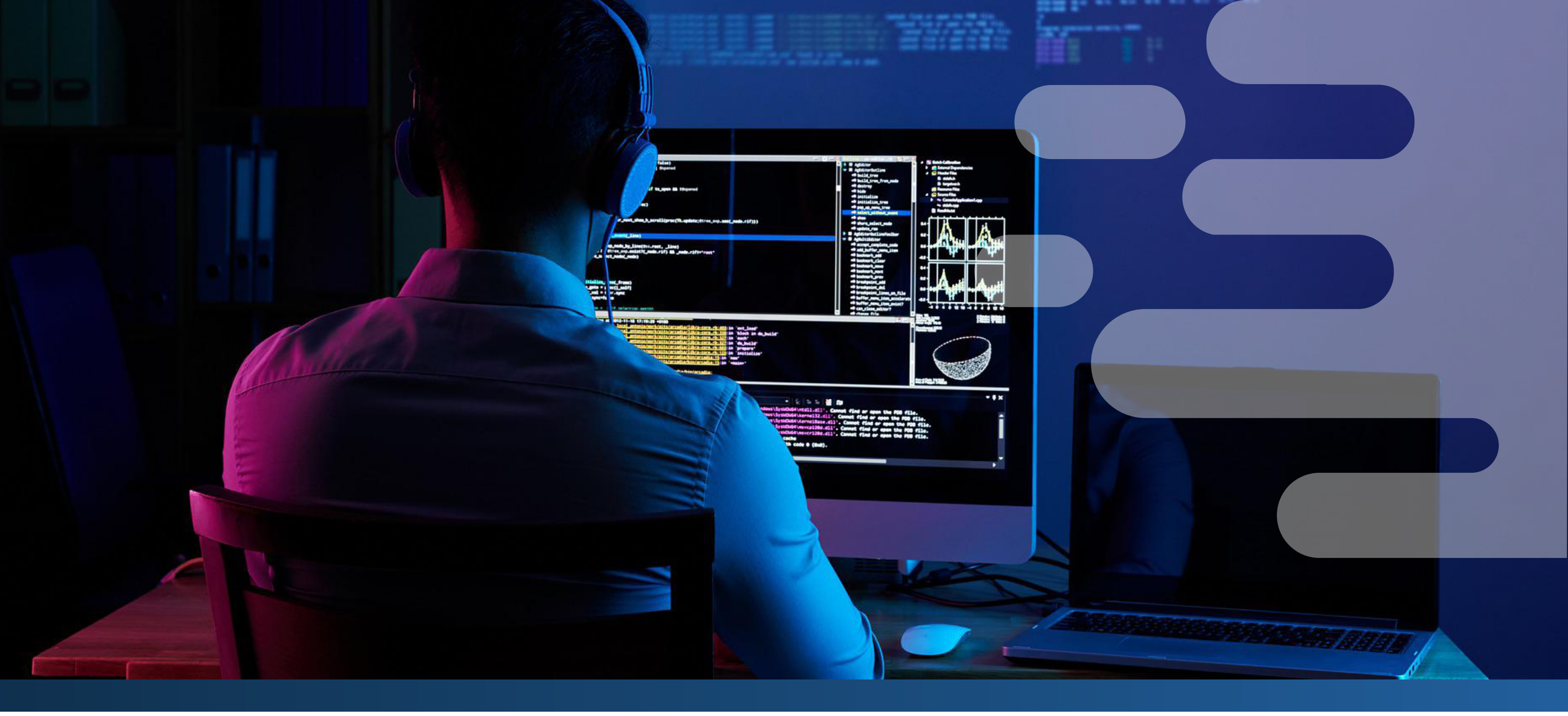
There are five (5) stages in a developer's journey to purchase a product:

Discover	Evaluate	Learn	Build	Scale
01	02	03	04	05
What is the product and is it of use to me? Is it credible and does it solve my problems?	Will it meet my needs, is it easy to use, are there any red flags, and is pricing a barrier?	How does it work, time to HELLO WORLD, is there a community and good documentation to help me when I get stuck?	Can I build a proof of concept, is there a good experience, what is the support like, is it value for money?	Can I scale and do more, how do I give feedback and contribute, will the product grow with me and my company's needs?

Developer education moves the developer along this buying journey, going from initial awareness of the product through activation, downloading the software, getting an API key, and then engaging and using it regularly.

Developer education primes people to be receptive to your product and to understand why they would adopt it in the first place. It is about helping developers understand and experience the value of your product and the problems it solves. It's very hard to get someone to buy your product if they don't understand why they would use it. A lot of developer education is just helping developers understand the why (i.e. why does this product exist?), not just the what (i.e. what does this product do?).

If you don't have the ability to educate the developer, you'll lose a lot of them as they start evaluating the product. They'll sign up for your product and will disengage as soon as they realize that they need to attend a sales demo before they can trial your product. They will fall out of the funnel and will start looking at another product that provides a more developer-friendly learning experience.



Even if you provide an ungated free trial aimed at developers, you can still lose them if you don't have learning resources to explain how to use your product to solve their problems. Even a Hello World example is not enough because they still don't understand the product's value in helping them address their particular use case. Instead, developers want to play around with your product (and maybe even break it), and be guided along this process with useful tutorials, code examples, documentation, courses and labs.

Getting developers to do more with your product is a challenge, but each part of your developer education program should be like a stepping stone to the next until they reach the "Aha! Moment." The quicker you can get them there, the better. You don't have a lot of time to get them to this "Aha! Moment" (see quote), which means each element has to be effective and well-tested. You need to measure how well it's going and make improvements based on this data.

"If I don't see how the product is going to work for me within 20 minutes, it's probably not going to work for me."

– DevOps Engineer

What is the **current state** of developer education?

Current developer education initiatives often lean on copious amounts of documentation with ongoing “nurture” campaigns that follow traditional marketing techniques. This includes providing static fact sheets that talk about the product’s benefits, email workflows telling developers about the product, or videos that walk developers through the product’s high-level benefits.

Developers want to figure out your product and use it in real-life situations that they’ll encounter on a day-to-day basis. They want educational and technical product information that they can find in your documentation, as well as interactive, software sandboxes that they can find in a developer education platform.

So, instead of marketing to them or providing static experiences, you need to engage developers with an educational developer marketing initiative that helps them learn about your product. Providing developers with product education means they are more likely to feel fully educated on your product’s capabilities, its potential, and (perhaps more importantly) its limitations.

How does developer education differ from developer marketing?

Developer marketing and developer education have different purposes and audiences. Although there is often crossover, they both work together for the same ultimate goal.

Audience

Developer marketing initiatives mainly target decision-makers, the people who are going to be signing the check to buy the product. This may include both developers and the C-suite, as developers are increasingly influencing purchase decisions. According to research, 95% of developers have some role in purchasing, while 60% of developers can approve or reject a technology purchase.

On the other hand, developer education is primarily focused on end-users. Developer education targets the people who use the product on a day-to-day basis and who need to know how a product works. They need to see if the product makes their jobs easier or if the product fulfills a specific set of technology requirements specified by their company.

Goals

The aim of developer education is product adoption and usage, rather than purchase. Developer educators are focused on building competency and success. They want to empower developers who know how your product works and can use all of the features to build cool things.

Kasey Byrne who has led marketing at Postman, npm, Rasa, and now Botpress, explains the idea: “When you’re selling a developer product, the story needs to be ‘Use our product, and you’ll be able to build something really cool.’” She adds that the storyline needs to be front and center in your developer marketing content.

The education team wants to make sure people are learning about the technology, taking courses, getting hands-on experience, and accessing product documentation. Ultimately, developer education might lead to a purchase somewhere down the line, but it’s not the immediate aim of developer education.

Developer marketers want to generate leads and nurture developers through the entire purchasing journey until they buy the product. If users drop out along the way, that is an issue for them. Developer marketers also use marketing techniques to convert users on their free tiers – those who take courses, read documentation, and participate in product communities – into paying customers.

“When you’re selling a developer product, the story needs to be ‘Use our product, and you’ll be able to build something really cool.’”

– Kasey Byrne, Marketing Leader

What does a developer education program involve?

A developer education program aims to get the developer to engage deeper into your product. Forget about selling your product. Instead, aim to lower the friction between developers and your product. Before selling your product, you first need to engage developers with an educational experience that helps them learn about it.

As Ashley Smith, Former Global Vice President of Marketing at GitHub, says: "Almost everything we do is about teaching."

Below are the elements you need to create a successful developer education program. The purpose of these tools and techniques is to get developers to the point where they can see the value in your product and how it helps them solve their engineering problems.

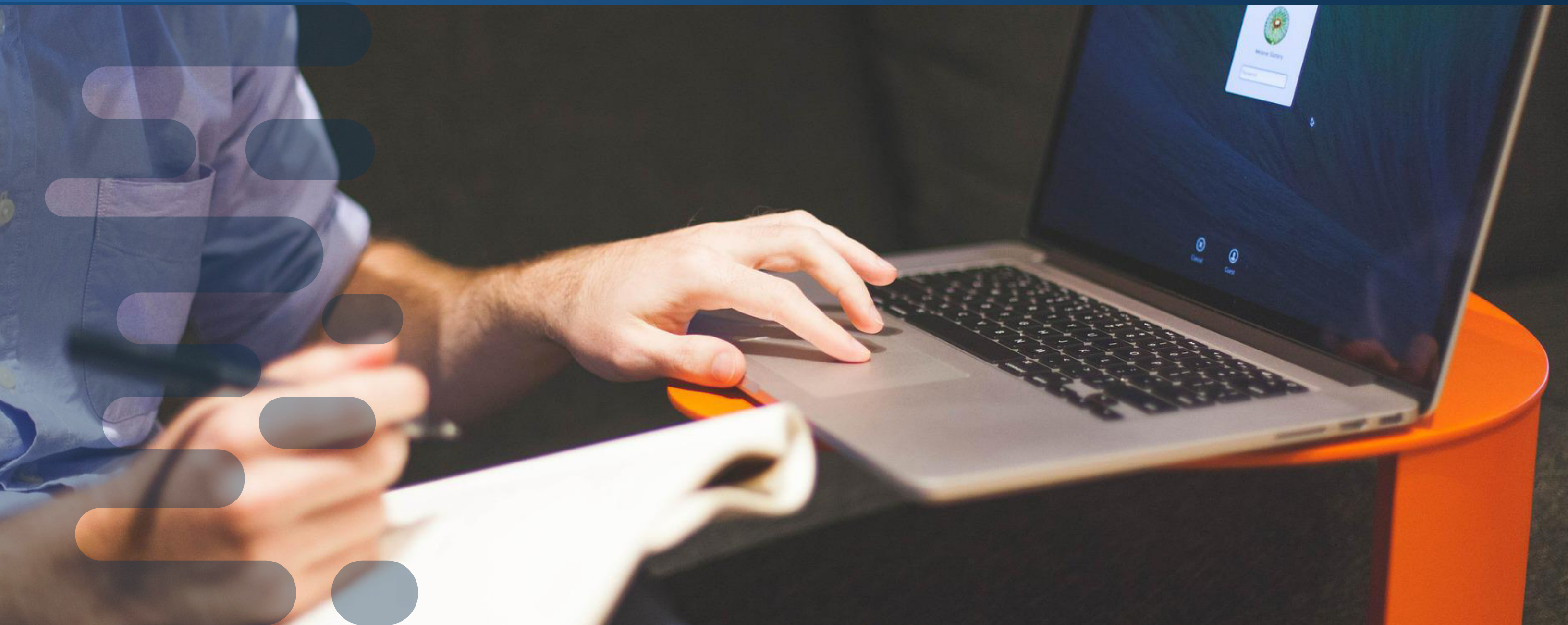
Industry content

Before you can get people to purchase your product, you need to educate them on the problem that your product solves, what the problem is costing them, how your product is helping other users, and the limitations of current solutions or processes.

A lot of developers don't even know they need your product; they're just trying to solve a problem. So if you only create content about your product (and not about the problem your product solves), you could miss a large population of developers who are good candidates to use or purchase your product. Similarly, if you only create product-related content, many developers will bounce because they aren't at the right stage of their purchasing journey (or aren't educated enough on the problem) to want to know more about your product.

It's important to provide tips, tutorials, and how-to guides with useful information, rather than content that solely focuses on your product. This way, if a developer is Googling keywords on a particular issue, they will find your content. And if you provide tutorials or guides that help solve their problem, they'll read those articles and will naturally become more curious about your company and what you do. Once they know about your product, you can then direct them towards more product-focused content.

Andrew Racine, VP of Global Demand Gen at Fivetran, explains how the top viewed posts on their blog didn't deal with their product. They all covered related industry trends and how-to's. The number one post on their blog is a comparison of MySQL and PostgreSQL. The content is peripherally related to their product, so it helps get more developers interested in Fivetran.



Self-paced courses

Once a developer knows about your product and the problem(s) it solves, you can start to help them to learn how to use your product. That's where self-paced training comes in, which can help companies deliver educational content that works for different learning styles and different types of developers with different roles and responsibilities.

You might already be hosting live workshops or conducting instructor-led training. While this form of training can be highly beneficial and interactive, there are a number of downsides, including:

- Instructor availability
- Timezone issues
- Limited class-size might exclude people
- Challenges catering content to different audiences and skill levels
- Inability for developers to rewind and consume content at their own pace
- Language issues if English isn't a developer's native language (compared to self-paced courses, where the videos can have multilingual subtitles)
- Limited reporting and tracking of developer outcomes (who was successful in completing the exercises and who's still stuck?)

With self-paced training, developers can revisit the content as often and as many times as they like as opposed to instructor-led training where if you miss the training, then you miss out on the content and don't get the benefits. If they find a particular module difficult, they can spend more time on it. If they find other modules easy or not relevant to their role, they can speed through them or skip them altogether.

These courses also allow you to collect a lot of granular data about the developer that isn't possible if you only provide blogs or static tutorials, which have no assessment capabilities. You can see where developers are getting stuck and where they are dropping off and use this data to create better courses and improve the developer's learning experience.

Developers also get real-time feedback as they go through the content, including knowledge checks and quizzes. They're validating their understanding of the product in an interactive way that reading a blog post can't. They can also see the clear progression towards the end of the course, helping to nudge them forward and to keep them motivated to learn more about your product.

Bite-sized content

Successful developer education programs provide bite-sized content that developers can easily complete, or skip if they already know how to do it. Developers are task-oriented and want to finish these bite-sized pieces of learning content before moving to the next level if they're still interested. But if you throw something at them that assumes they already have a range of different skills, or have in-depth knowledge about your product, it's not going to work. They're going to feel overwhelmed and go elsewhere.

You need to give them bite-sized morsels so you keep them engaged and wanting to learn more about your product. Every time they finish one of your developer learning experiences, you need them to want to move to the next level and learn more so they can become an advocate for the adoption or purchase of your product.

Personalized learning paths

There is no one-size-fits-all education program, just as there is no one typical developer. Different developers have different learning styles. Some might want to learn face-to-face with an instructor, others will want access to self-paced courses or to watch videos and use tutorials.

Instead of taking a one-size-fits-all approach, your developer education programs should include personalized learning paths that are tailored around specific roles, product features, career paths, programming languages, or use cases. For example, if your product is used to build many different things, you want content that is tailored to particular use cases that developers will be working on. Or, you might develop a special course for a Python developer that's different from the course for a Java developer.

To be able to personalize your content according to the above criteria, you need to create buyer personas to understand who you're targeting and what they need. To build these personas, you can use surveys to understand developer demographics and work out who the developer is, including their goals, background, and skill level.

If you provide developers with personalized learning paths that work for their personas that they can complete in their own time, you will help overcome any time constraints, reduce the friction between your product and developers, and enable them to test your product in a way and at a time that suits them.

A sandbox environment

When a developer comes to your website, they aren't going to read or even believe most of the marketing content on your product. They want to see how your product works, to play around under the hood and make their minds up about whether it does what they need. And they can't do this in one sales-led demo.

A developer needs a sandbox environment to try out your product early on in the purchase journey. They need interactive product learning experiences that let them try out your product in real-world scenarios (rather than sitting through a demo). Hands-on software sandboxes will also ensure that they get a learn-by-doing environment that lets them analyze your product to determine if it solves the problem they're working on, or delights them enough to justify using or buying your product.

Interactive documentation

Once a developer has spun up a sandbox, you can use other educational content to help them better understand how it works and make sure they are using all the features to the fullest. By turning your existing developer documentation into immersive, educational experiences, you can improve developer growth, developer adoption, product usage, API calls, and built applications.

To educate themselves about your product, developers need to read copious amounts of uninterrupted, continuous documentation. Consider adding interactivity into your developer documentation with:

- Live API calls
- Videos
- FAQs
- Discussion forums
- Hands-on sandbox environments
- Sample code

A tailored developer zone

It's important to build a dedicated developer zone that speaks their language and contains all of your developer-first, technical product content. Creating a separate and dedicated "developer zone" is important because your corporate website will have a different tone (that likely caters to a different audience), so having a dedicated developer page will allow you to present technical product content in the tone and detail that developers expect.

Here are some developer zones you can get inspiration from:

- Learn Chef
- Hummingbot
- Dremio
- Redis

A developer community

Building developer communities is an important aspect of developer education. Developers often choose products based on recommendations from their peers that they receive in communities. Developers talk to each other about issues such as which products work, which products helped them scale, and what the product's developer support was like. If you're a developer-first company, you want to be a part of that community and have a first-tier reputation for supporting your developers. Even when the developer community's feedback is negative, you want to hear what they are saying and understand their concerns.

If you can create a community around your product yourself, that works even better. Giving developers the ability to ask each other questions about your product builds trust and credibility; it shows you have nothing to hide. Enabling peer-to-peer discussions creates transparency about your product's capabilities and limitations because developers have higher confidence in their peers' first-hand experience with your product than what your marketing collateral says.

What metrics do you use to measure developer education success?

Developer education metrics differ from both traditional marketing metrics and overall developer marketing metrics. Purchase rate is not the main metric most companies measure. Instead, they focus on usage and adoption.

A lot of products are sold from the bottom up. A developer starts using your product for free. Then they start using your product regularly, invite coworkers, then graduate to writing their code around your product. These are the important metrics to measure. In the early stages of the developer journey; it's all about activity and product engagement (not developer monetization).

In the early stages of this process, getting developers to engage with your product regularly is key. You can figure out how to monetize them later. If a developer is already using your product every day, knows how it works, and can see the benefits, then they are more likely to upgrade to a subscription – whether that be through an enterprise license, a support and training package, hosting, or something else.

Below are some of the main metrics to measure when it comes to determining the success of your developer education program:

Adoption

The following indicators play into your adoption rate:

- How many virtual labs or software sandboxes are being spun up
- How many trials (attributed to your B2D efforts) are being spun-up
- How many applications are being built or have been built
- Number of applications per developer
- Number of 3rd party integrations onto other platforms

Engagement

Engagement rate will require you to look at:

- How often are developers using your product
- How often specific features within a product are being used
- Number of API calls (assuming your company sells or markets an API)
- Average number of logins per developer
- Daily active users (DAUs)
- Number of developers engaging with your product 30 days after sign-up

Using developer education to shape your product

Educating developers about your product is a form of customer discovery. Creating educational content also helps you to refine your ideal customer profile (ICP), better understand your customers, and tailor your messaging accordingly.

Creating a community is a key part of developer education. By creating this community, you can find product champions who will be evangelists for you within their networks as well as clusters of users who you might be able to convert further down the line. The feedback and information you get from this community can shape your product roadmap. You can learn more about your product, its strengths and weaknesses, where it adds value, and which areas you should create educational content to help developers better understand how your product works.

Throughout every stage of building a company, you have to learn from your users, understanding what they are going to find most valuable, and what is going to bridge that gap to an experience that is much better than the one they currently have. It's about shaping your product to meet your audience's real problems, not what you think they might be struggling with. Creating educational content that matches the needs of your developer audience is crucial for every phase of building a developer-first, software product.

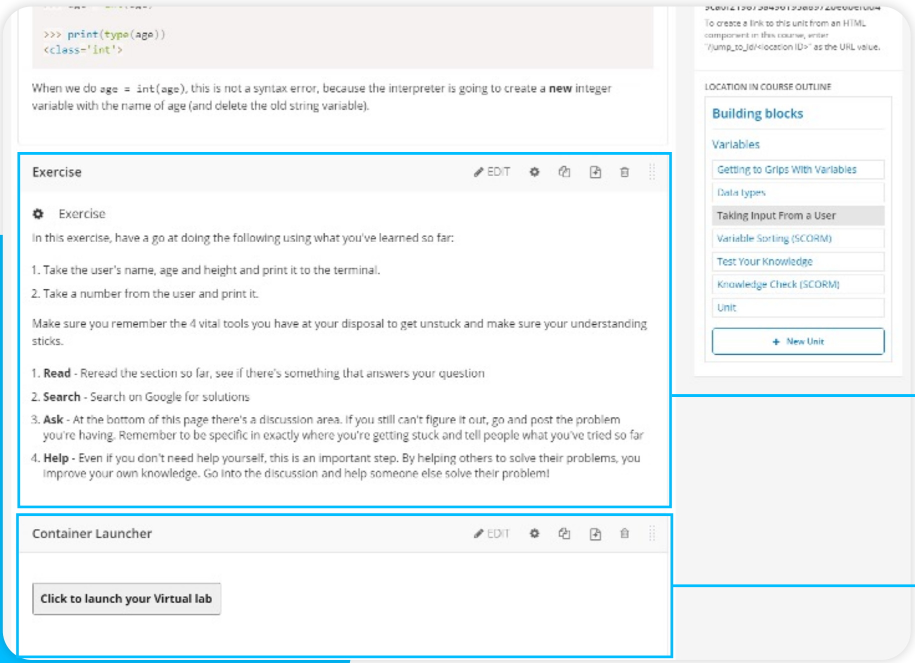
Developer education in action

By now, you might be wondering what a successful developer education program looks like in practice. Here are some examples of companies that are reaping the benefits and getting tangible results from their developer education programs.

Chef Software

Chef Software's DevOps automation tools enable enterprises to overcome the complexities involved with automating their infrastructure, security, systems, and applications. The company's developer education program has helped improve product awareness and adoption.

To use Chef's tools, you need to know Ruby and have knowledge of DevOps. With that in mind, Chef created courses through its Learn Chef online university that teach developers the fundamental skills you need to be successful not just with their products, but with many other tools and software.



```
>>> print(type(age))
<class='int'>
```

When we do `age = int(age)`, this is not a syntax error, because the interpreter is going to create a **new** integer variable with the name of age (and delete the old string variable).

Exercise

Exercise

In this exercise, have a go at doing the following using what you've learned so far:

1. Take the user's name, age and height and print it to the terminal.
2. Take a number from the user and print it.

Make sure you remember the 4 vital tools you have at your disposal to get unstuck and make sure your understanding sticks.

1. **Read** - Reread the section so far, see if there's something that answers your question
2. **Search** - Search on Google for solutions
3. **Ask** - At the bottom of this page there's a discussion area. If you still can't figure it out, go and post the problem you're having. Remember to be specific in exactly where you're getting stuck and tell people what you've tried so far
4. **Help** - Even if you don't need help yourself, this is an important step. By helping others to solve their problems, you improve your own knowledge. Go into the discussion and help someone else solve their problem!

Container Launcher

To create a link to this unit from an HTML component in this course, enter "/jump_to_id/<location ID>" as the URL value.

LOCATION IN COURSE OUTLINE

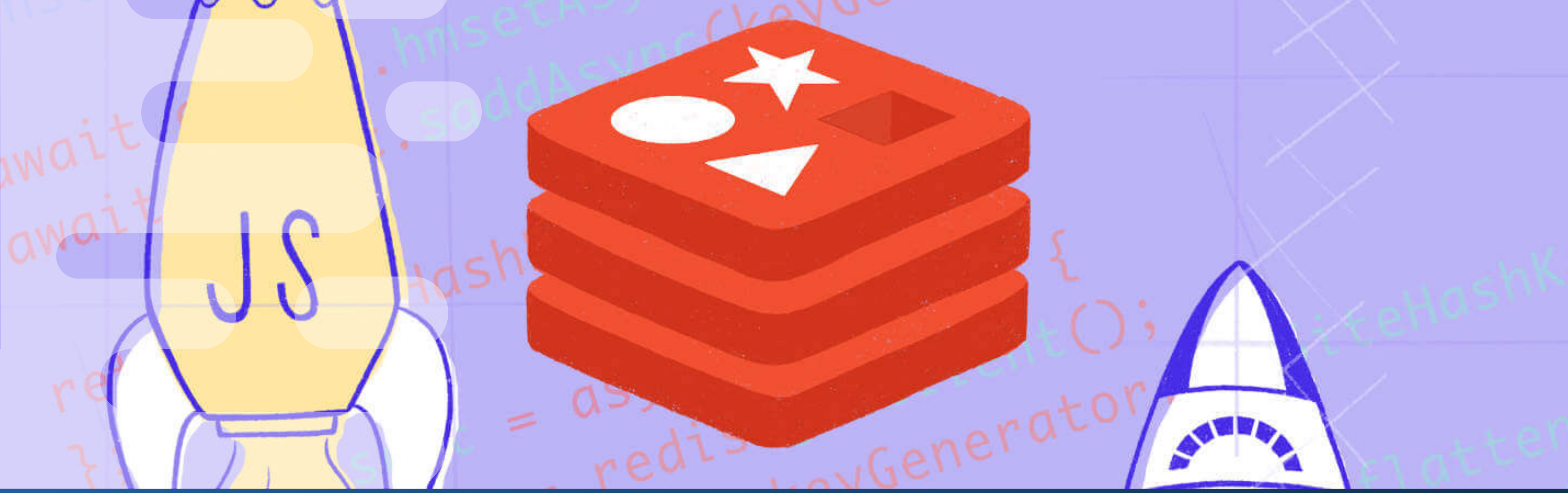
Building blocks

Variables

- Getting to Grips With Variables
- Data types
- Taking Input From a User
- Variable Sorting (SCORM)
- Test Your Knowledge
- Knowledge Check (SCORM)
- Unit

Educational content is given to the learner to provide context and learning principles about the product.

After reading the content above, the learner launches the product for hands-on learning.



When people are searching for resources to learn Ruby, for example, Chef's courses appear on the results page. When developers are finding out more about the course, or when they enroll and start learning about Ruby, they can also investigate the company and the product to see if it looks useful for them.

And once a developer gains interest in their products, Chef uses hands-on product sandboxes to enable developers to spin up their product quickly. Users don't have to install anything on their local machine or worry about setting up a virtual box and which operating system they're using, which lowers the barrier to usage.

Redis

Redis is a real-time data platform, which delivers unmatched performance, scalability, innovation, and cost-effectiveness across cloud, on-premise, and hybrid deployments. Despite more than 1.4 billion downloads of their software, they didn't have these users' contact information and had no idea who they were. Redis had no way of making a connection with the anonymous developer who downloaded their software.

To solve this problem, they created the Redis University where they reached out to Redis developers and invited them to take free courses. In exchange for getting these free courses, the developers needed to register and provide an email address. Once Redis had those email addresses, it opened up many possibilities, including looking for clusters of users who worked at the same company. These clusters were then passed to Redis' sales team, which signaled an account with high purchase intent.

Redis also provided their users with an easy way to complete hands-on, software exercises during online training and removed the friction between Redis' learners and their products.

Why don’t traditional learning management systems work for developer education?

If you’re using a traditional learning management system (LMS), it can be a cumbersome process to create the hands-on, educational experience that developers need to adopt your product.

LMS often focus on video or multiple choice-based content and assessments, which isn’t an effective way to teach developers about a technical product. When someone is learning how to code or how to use technical software, they need to get their hands on the product and learn by doing. Most traditional LMSs are not set up to enable hands-on learning and, in fact, sometimes add more friction between developers and your product. LMSs are an antiquated way of delivering technical training and are unsuitable for developer education.

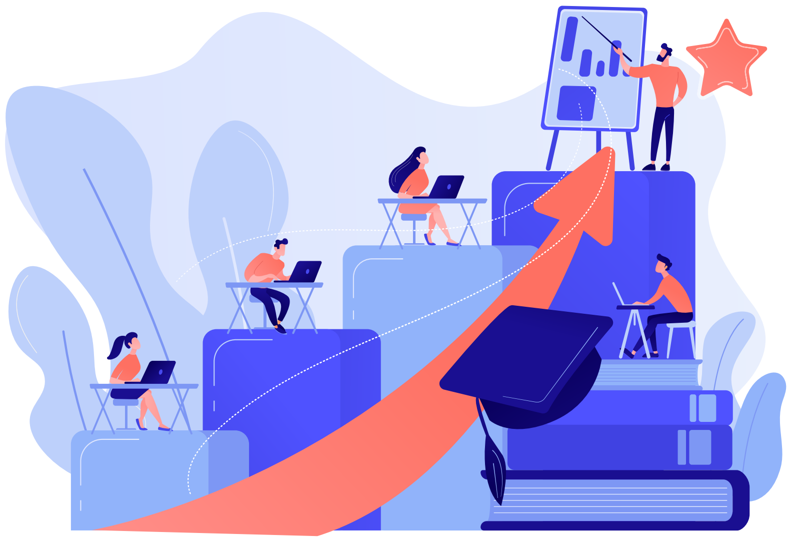
What companies need for developer education is content that is more integrated with the product experience. They need to be able to drop developers directly into the product, show them its value by experiencing how it works, learn by making mistakes, and create those “A-ha moments!” when they get it right. But you can't just drop people in a product and then walk away and expect them to be successful with it. You need an educational platform that launches a hands-on product experience next to educational content.

What is a developer education platform?

A developer education platform enables you to build frictionless and educational product experiences for developers. This approach provides genuine, educational content about your product that lets developers experience its features through hands-on learning, experimenting with it, and figuring out how it works.

A developer education platform combines a learn-by-doing approach that includes hands-on sandboxes with self-paced courses to provide the educational experience developers need to decide whether to adopt or purchase your product.

By creating your product courses on a developer education platform, you can reinforce and emphasize the more important, most visited, or most asked-about sections of the developer documentation. And as developers become more comfortable with how your product works, they will adopt your product and begin writing code around it. Over time, it will encourage them to share your product with their community, with other developer communities, and graduate into an active community participant in helping other developers learn and adopt your product.



appsembler

Build it with education.

Speak to Appsembler

Start a conversation with Appsembler to discuss examples of modern, developer education initiatives.

Contact Us